МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по практической работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: Стек

Студент гр. 8304		Бочаров Ф.Д.
Преподаватель		Фирсов М.А.

Санкт-Петербург
2019

Цель работы

Решить задачу, используя стек, на базе списка. Получение навыков работы с нелинейными структурами данных.

Задание

В заданном текстовом файле F записано логическое выражение (ЛВ) в следующей форме:

<ЛВ>::= *true* | *false* | (¬< ЛВ >) | (< ЛВ > ^ < ЛВ >) | (< ЛВ > ˅ < ЛВ >)

Описание алгоритма

Программа проверяет строку на корректность, далее алгоритмом форматирует ее в постфиксную запись. После чего благодаря работе со стеком вычисляет истинность введенного выражения

Описание основных функций

void push(char new_val) – добавляет элемент в стек

char pop() – поднимает элемент из стека

void is_correct(std::string str, int size) – проверяет на корректность

Вывод.

Были получены навыки работы с абстрактными типами данных, в частности со стеками.

ПРИЛОЖЕНИЕ А

Тестирование программы

Входные данные	Выходные данные
(!f)	t
(fVt)	t
((fVt)V(f^t))	t
((!f)^(fVt))	t

ПРИЛОЖЕНИЕ Б

Файл main.cpp

```
#include <iostream>
#include <string>

class Stack
{
public:
     struct Element{
        char value;
        struct Element* next;
        Element(char new_val = 0): value(new_val), next(nullptr){}
        ~Element(){
            delete next;}
    }elem;

    Stack(): Head(nullptr) {}

    ~Stack() {
        delete Head;
    }

    bool isEmpty() {
        return Head == nullptr;
    }

    void push(char new_val) {
        if(Head == nullptr) {
            Head = new Element(new_val);
        }
        else {
            Element* tmp = Head;
            while(tmp->next != nullptr) {
                tmp = tmp->next;
            }
            tmp->next = new Element(new_val);
        }
    }

    char pop() {
        if(Head == nullptr) {
            std::cout << "Is empty" << std::endl;
```

```
            return 0;
        }
        Element* tmp = Head;
        char val_ = Head->value;
        Head = Head->next;
        tmp->next = nullptr;
        delete tmp;
        return val_;
    }
    void is_correct(std::string str, int size)
    {
        int c, countSim = 0, countBracet = 0;
        for(c=0;c<size;c++)
        {
            if(str[c] == '(' || str[c] == ')')
            {
                std::cout << "Is incorrect" << std::endl;
                exit(1);
            }

            if(str[c] == 'f' || str[c] == 't')
            {countSim++;}
            if(str[c] == 'v' || str[c] == '^')
            {countSim--;}

        }
        if(countSim != 1)
        {
                std::cout << "Is incorrect" << std::endl;
                exit(1);
        }

    }


    private:
    Element* Head;

};
```

```
int main(int argc, const char * argv[]) {
    std::string str, result;
    char operand[100];
    int indOper = 0, indStr, bracketcount = 0;
    int size;
    std::cout << "expr?" << std::endl;
    std::cin >> str;
    size = str.size();

    for(indStr = 0; indStr < size; indStr++)
    {
        if(str[indStr] == '('
        {bracketcount++;}
        if(str[indStr] == ')')
        {bracketcount--;}
        if(str[indStr] == '(' || str[indStr] == '^' || str[indStr] == 'v' ||
str[indStr] == '!'){
            operand[indOper] = str[indStr];
            indOper++;
        }
        if(str[indStr] == 'f' || str[indStr] == 't'){
            result += str[indStr];
        }
        if(str[indStr] == ')'){
            result += operand[indOper-1];
            indOper-=2;
        }
    }
    if(bracketcount != 0){
        std::cout << "Is incorrect" << std::endl;
    exit(1);}
    size = result.size();
    Stack point;
    char a,b;
    point.is_correct(result, size);
    for(indStr = 0; indStr < size; indStr++)
    {
        if(result[indStr] == 't' || result[indStr] == 'f'){
            point.push(result[indStr]);
        }
```

```
        if(result[indStr] == '!'){
            a = point.pop();
            if (a == 'f'){
                point.push('t');}
            else
                point.push('f');
        }
        if(result[indStr] == '^'){
            a = point.pop();
            b = point.pop();
            if(a == 't' && b == 't')
                point.push(a);
            else(point.push('f'));

        }
        if(result[indStr] == 'v'){
            a = point.pop();
            b = point.pop();
            if(a == 't' || b == 't')
                point.push('t');
            else(point.push('f'));

        }

    }




  //  std::cout << "--" << std::endl << result <<std::endl;
    std::cout << "== " << point.pop() << std::endl;

    return 0;
}
```